B: Amendments to The Claims:

Amend the claims to read as follows:

Claim 1. (Currently Amended) A multiprocessor computer system, comprising:

a plurality of processing nodes and a plurality of dynamic cache coherency regions using caches associated with said processing nodes, and having

cache controller logic in said processing nodes controlling ~~movement of~~ software-initiated movement of software processes between said plurality of cache coherency regions by changing coherency mode bits without requiring a selective purging of cache contents in one or more of said processing nodes and when moving a software process between two distinct sets of processing nodes a cache line cannot be marked as shared in two separate coherency regions, and when said supervisor program is moving a coherency region from one distinct set of processing nodes to another distinct set of processing nodes it is effectively leaving behind cache entries for the coherency region on the old nodes and ensures that these old cache entries will be seen by incoming storage requests originating from the new processing nodes and that cache entries for the same main storage addresses will not be established in the new processing nodes until the old entries are invalidated.

Claim 2. (Currently Amended) The multiprocessor computer system according to claim 1, including supervisor software which enables said cache controller logic in a processing node to be sure upon an incoming storage request of a processor for a storage address that no copy of the requested storage address exists outside that processor's

current coherency region, as specified by a ~~the~~ current coherency region mode, whenever a cache entry for a requested storage address is found to exist on any cache in said processing nodes that contains ~~a~~ the processor that initiated said incoming storage request.

Claim 3. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region.

Claim 4. (Original) The multiprocessor computer system according to claim 2, wherein supervisor software creates a table for each processing node in the system which has an entry for every Coherency Region ID that is currently allowed to be dispatched on said processing node.

Claim 5. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region and one or more coherency mode bits for each processor in the multiprocessor computer system, and said coherency mode bits and coherency region ID associated with a processor are sent together with each storage transaction that is initiated by that processor when the transaction is transmitted for communication to another processor of said multiprocessor computer system.

Claim 6. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region and one or more coherency mode bits for each processor in the multiprocessor system to

to a node controller for a processing node.

Claim 7. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region and one or more coherency mode bits for each processor in the multiprocessor computer system, and wherein said mode bits associated with each transaction are used to determine which caches must participate in any storage transactions that they receive from any of the processors of said multiprocessor computer system.

Claim 8. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software creates a unique Coherency Region ID for each process that has its own coherency region and one or more coherency mode bits for each processor in the multiprocessor computer system and enables multiple cache coherency regions to operate without the use of cache purges during some operations which move software processes between coherency regions.

Claim 9. (Currently Amended) The multiprocessor computer system according to claim 8, wherein said supervisor software moves a software process out of one coherency region that is no longer going to be used by said software process and into another ~~software process~~ coherency region that has been created to cover the same address space as the first but which will include a new set of processing nodes.

Claim 10. (Original) The multiprocessor computer system according to claim 2, wherein said supervisor software

creates a unique Coherency Region ID for each process that has its own coherency region and moves a software process from one coherency region encompassing one set of processing nodes to another coherency region encompassing another set of processing nodes without requiring cache purges of caches in any of the processing nodes.

Claim 11. (Original) The multiprocessor computer system according to claim 10, wherein if said another coherency region contains fewer hardware processing nodes than the original coherency region, the size of the coherency region for said processing nodes has been effectively been reduced.

Claim 12. (Currently Amended) The multiprocessor computer system according to claim 1, wherein said multiprocessor computer system having a plurality of said ~~processing~~ processing nodes uses a table of active coherency region information associated with each processing node to determine when to alter the processing nodes' cache state transitions.

Claim 13. (Original) The multiprocessor computer system according to claim 12, wherein a supervisor program initializes said tables associated with each processing node and an entry in said table is made for each coherency region that the supervisor program intends to use on that processing node.

Claim 14. (Original) The multiprocessor computer system according to claim 1, wherein a supervisor program assigns a unique coherency region ID for each coherency region which the supervisor can associate with all software processes

that access storage addresses that are encompassed by the coherency region.

Claim 15. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node.

Claim 16. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node to thereby identify cache lines that are no longer actively used by any software processes on that processing node and to change the cache entries for that processing node to invalid in response to a storage request from outside the coherency region.

Claim 17. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node and allows all of the caches in said multiprocessor computer system to continue processing coherency transactions while the coherency boundaries for a software process are effectively changed.

Claim 18. (Original) The multiprocessor computer system according to claim 1, wherein processing nodes are able to

identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node such that cache lines belonging to a software process that is no longer actively being dispatched on a given processing node are identified and invalidated thereby enabling their reuse.

Claim 19. (Original) The multiprocessor computer system according to claim 1, wherein a supervisor software uses processor state information to determine which caches in the multiprocessor computer system are required to examine a coherency transaction produced by a single originating processor's incoming storage request.

Claim 20. (Currently Amended) The multiprocessor computer system according to claim 19, wherein a processing node of the multiprocessor computer system ~~has~~ has dynamic coherency boundaries such that the multiprocessor computer system uses only a subset of the total processors in a system for a single workload at any specific point in time and can optimize the cache coherency as the supervisor software expands and contracts the number of processors which are being used to run any single workload.

Claim 21. (Original) The multiprocessor computer system according to claim 20, wherein multiple instances of processing nodes can be connected with a second level controller to create a large multiprocessor system.

Claim 22. (Original) The multiprocessor computer system according to claim 21, wherein said supervisor software creates a unique Coherency Region ID for each process that

has its own coherency region and one or more coherency mode bits for each processor in the multiprocessor computer system and enables multiple cache coherency regions to operate without the use of cache purges during some operations which move software processes between coherency regions and a node controller uses said mode bits to determine which processors must receive any given transaction that is received by the node controller.

Claim 23. (Original) The multiprocessor computer system according to claim 22, wherein a second level controller uses the mode bits to determine which processing nodes must receive any given transaction that is received by the second level controller.

Claim 24. (Original) The multiprocessor computer system according to claim 21, wherein said supervision software uses logical partitions which are mapped to allowable physical processors and a distinct cache coherency region can be defined for each partition using a hypervisor.

Claim 25. (Original) The multiprocessor computer system according to claim 2, wherein said coherency region ID is used to perform the function of a cache coherency mode and a node controller determines which physical processing nodes are associated with specific coherency region Ids.

Claim 26. (Original) The multiprocessor computer system according to claim 3, wherein any incoming storage request which misses all of the caches in an originator's coherency region is then sent on to all processing nodes in the entire system, regardless of the setting of said mode bits.

Claim 27. (Currently Amended) The multiprocessor computer system according to claim 3, wherein any incoming storage request which requests ~~Requests~~ which hit in an originator's coherency region but which do not have a correct cache state do not need to be sent outside the coherency region.

Claim 28. (Currently Amended) A method for use in a multiprocessor computer system, comprising the steps of:

moving software processes between a plurality of cache coherency regions for caches associated with a plurality of processing nodes of said multiprocessor computer system without requiring a selective purging of cache contents in one or more of said processing nodes, after supervisor software creates a unique Coherency Region ID for each process that has its own coherency region, and

said supervisor software creates a table for each processing node in the ~~multiprocessor~~ multiprocessor computer system which has an entry for every Coherency Region ID that is currently allowed to be dispatched on said processing node.

Claim 29. (Original) The multiprocessor computer system according to claim 28, wherein said coherency mode bits and coherency region ID associated with a processor are sent together with each storage transaction that is initiated by that processor with a requested storage address when the transaction is transmitted for communication to another processor of said multiprocessor computer system.

Claim 30. (Original) The multiprocessor computer system according to claim 28, including a step of enabling with supervisor software the multiprocessor computer system's cache controller logic in a processing node to be sure that,

upon receipt at said processing node that an incoming storage request for a storage address, no copy of the requested storage address exists outside said processing node's current coherency region, as specified by a current coherency region mode, whenever a cache entry for a requested storage address is found to exist on any cache in any of said multiprocessor computer system's processing nodes that contains a processor that initiated said incoming storage request.

Claim ~~27~~ 31. (RENUMBERED AND CANCELED) ~~The multiprocessor computer system according to claim 3, a cache line cannot be marked as shared in two separate coherency regions, and when said supervisor program is moving a coherency region from one distinct set of processing nodes to another distinct set of processing nodes it is effectively leaving behind cache entries for the coherency region on the old nodes and ensures that these old cache entries will be seen by incoming stroage requests originating from the new processing nodes and that cache entries for the same main storage addresses will not be established in the new processing nodes until the old entries are invalidated.~~